

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P. O. Box 272400
Fort Collins, Colorado 80527-2400

IFW
AE
PATENT APPLICATION

ATTORNEY DOCKET NO. 200308325-1

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): TODD DAVID MILLSTEIN et al

Confirmation No.: 4747

Application No.: 09/754890

Examiner: Michael Yigdall

Filing Date: Jan 05, 2001

Group Art Unit: 2122

Title: SYSTEM & METHOD FOR FOR VERIFYING COMPUTER PROGRAM CORRECTNESS AND
PROVIDING RECOVERABLE EXECUTION TRACE INFORMATION

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Sir:

Transmitted herewith in triplicate is the Appeal Brief in this application with respect to the Notice of Appeal filed on August 10, 2004.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) \$330.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

() (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d) for the total number of months checked below:

() one month	\$110.00
() two months	\$420.00
() three months	\$950.00
() four months	\$1480.00

() The extension fee has already been filled in this application.

(X) (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account 08-2025 the sum of \$330.00. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

(X) I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450. Date of Deposit: August 17, 2004

OR

() I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number _____ on _____

Number of pages:

Typed Name: Be Henry

Signature: Be Henry

Rev 05/04 (Aptbrief)

Respectfully submitted,

TODD DAVID MILLSTEIN et al

By Philip S. Lyren

Philip S. Lyren

Attorney/Agent for Applicant(s)

Reg. No. 40,709

Date: August 17, 2004

Telephone No.: 281 514 8236



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Leino et al.	Examiner:	Michael J. Yigdall
Serial No.:	09/754,890	Group Art Unit:	2122
Filed:	January 5, 2001	Docket No.:	200308325-1
Title:	System and Method for Verifying Computer Program Correctness and Providing Recoverable Execution Trace Information		

APPEAL BRIEF UNDER 37 C.F.R. § 1.192

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This Appeal Brief is filed in response to the Final Office Action mailed May 17, 2004 (Paper No. 8) and the Notice of Appeal filed on August 10, 2004.

AUTHORIZATION TO DEBIT ACCOUNT

It is believed that no extensions of time or fees for net addition of claims are required, beyond those, which may otherwise be provided for in documents accompanying this paper. However, in the event that additional extensions of time are necessary to allow consideration of this paper, such extensions are hereby petitioned under 37 C.F.R. § 1.136(a), and any fees required therefore (including fees for net addition of claims) are hereby authorized to be charged to Hewlett-Packard Development Company's deposit account no. 08-2025.

08/20/2004 ZJUHR1 00000054 082025 09754890

01 FC:1402 330.00 DA

REAL PARTY IN INTEREST

The real party-in-interest is the assignee, Hewlett-Packard Company, a Delaware corporation, having its principal place of business in Palo Alto, California.

RELATED APPEALS AND INTERFERENCES

There are no known related appeals or interferences known to appellant, the appellant's legal representative, or assignee that will directly affect or be directly affected by or have a bearing on the Appeal Board's decision in the pending appeal.

STATUS OF CLAIMS

Claims 1 - 47 stand finally rejected. No claims have been allowed. The final rejection of claims 1 - 47 is appealed.

STATUS OF AMENDMENTS

An After Final (AF) Amendment was filed on June 24, 2004 subsequent to the final rejection. In the AF Amendment, four claims (11, 17, 39, and 47) were amended to correct for typographical errors.

In an Advisory Action dated August 2, 2004, the Examiner entered the AF Amendment. The claims in the Appendix correspond to the claims submitted in the AF Amendment and entered by the Examiner in the Advisory Action.

SUMMARY OF THE INVENTION

A summary of the invention is provided below with reference numerals and references to the specification and drawings. The summary is set forth in one exemplary embodiment as the language corresponding to independent claim 1. Discussions about elements of claim 1 can be found at least at the cited locations in the specification and drawings.

One exemplary embodiment includes a method (Figs. 2 and 3) of verifying with static checking (Fig. 1, 120) whether a specified computer program (Fig. 1, 118) satisfies a predefined set of conditions (Page 3, lines 31-32). The method includes converting the program into a logical equation representing the predefined set of conditions as applied to

instructions and elements of the instructions of the program (Page 7, lines 1-4). The converting step includes inserting flow control labels into the sub-equations of the logical equation (Page 7, lines 15-24). The flow control labels identify conditional branch points in the specified computer program (Page 9, line 25 to Page 10, line 17). A theorem prover (Fig. 1, 126) is applied to the logical equation to determine the truth of the logical equation (Page 7, lines 30-32). When the truth of the logical equation cannot be proved (Page 7, line 32 to Page 8, line 10), generating at least one counter-example (Fig. 1, 128) that identifies one of the conditions (Page 8, lines 12-19), one or more variable values inconsistent with the one condition (Page 8, lines 12-19), and any of the flow control labels for conditional branch points of the program associated with the identified variable values (Page 8, lines 12-19). At least one counter-example is converted into an error message (Fig. 1, 132) that includes a program trace that identifies a path through the computer program when the counter-example identifies one or more of the flow control labels (Page 8, lines 21-23; Page 8, lines 27-30; and Page 9, lines 7-10).

ISSUES

1) Whether claims 1-5, 8, 11, 14, 17, 20, 22-27, 30, 33, 36, 39, 42, 45, and 47 (rejected under 35 USC § 103(a)) are unpatentable over “Extended Static Checking” by Detlefs et al. (hereinafter Detlefs) in view of USPN 4,920,538 (Chan).

2) Whether claims 6, 7, 9, 10, 12, 13, 15, 16, 18, 19, 21, 28, 29, 31, 32, 34, 35, 37, 38, 40, 41, 43, 44, and 46 (rejected under 35 USC § 103(a)) are unpatentable over Detlefs in view of Chan as applied to claims 1-5, 8, 11, 14, 17, 20, 22-27, 30, 33, 36, 39, 42, and 45 above, and further in view of USPN 5, 854,924 to Rickel et al. (hereinafter Rickel).

GROUPING OF CLAIMS

The claims do not stand or fall together. Instead, the claims are grouped as follows:

Group I: Claims 1-46. Claim 1 is representative of the group.

Group II: Claim 47. Claim 47 is representative of the group.

The claims in Groups I and II are separately patentable. Claim 47 (the representative of Group II) recites numerous limitations that are not present in claim 1 (the representative of Group I). Further, claim 1 recites numerous limitations that are not present in claim 47. For example, claim 47 recites the following limitations that are not present in claim 1: “wherein at least one of the flow control labels is of a form selected from the group consisting of (1) $L \Rightarrow P$ wherein L is a flow control label name and P is a subcomponent of the logical equation, (2) $L = K \Rightarrow P$ wherein L is a flow control label name, k is a constant value and P is a subcomponent of the logical equation, (3) $\{L \text{BLPOS } L \text{ } P\}$ wherein L is a flow control label name and P is a subcomponent of the logical equation, (4) $\neg\{L \text{BLNEG } L \text{ } \neg P\}$ wherein L is a flow control label name and P is a subcomponent of the logical equation, and (5) $\{L \text{BLPOS } L \text{ True}\} \Rightarrow P$ wherein L is a flow control label name and P is a subcomponent of the logical equation.” The differences between recitations in claims 1 and 47 present different issues regarding patentability over Detlefs, Chan, and Rickel. These different issues are discussed in detail in the Argument section for Groups I and II.

ARGUMENT

A. GROUP I (Claim 1):

Claim 1 is representative of Group I. Claim 1 is rejected under 35 USC § 103(a) as being unpatentable over Detlefs in view of Chan.

To establish a prima facie case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art cited must teach or suggest all the claim limitations. See M.P.E.P. § 2143. Applicants assert that the rejection does not satisfy these criteria.

Detlefs and/or Chan not teach/suggest Tracing Path Limitation

Claim 1 recites converting at least one counter-example into an error message. The error message “includes a **program trace that identifies a path through the computer program** when the counter-example identifies one or more of the flow control labels” (emphasis added). This limitation is not taught or suggested in Detlefs and/or Chan.

The Examiner states that Detlefs teaches an error message that includes a program trace that identifies a path through the computer program when the counter-example identifies one or more of the flow control labels. The Examiner cites Detlef (Page 29, section 6, paragraph 2, lines 3-4 and 7-9) and Chan (Col. 1, lines 26-35).

This section of Detlef is reproduced below:

If the prover finds a counterexample, it emits the set of labels of relevant subformulas that are false in the counterexample. The name of the label encodes the source position and error type. This makes it straightforward to translate failed proofs into specific error messages.

This section of Chan is reproduced below:

One of the problems that arises in trying to check entire microcode sequences is checking the execution of conditional branching instructions. This is because branches outside the path taken in calculating the original check key will result in different check keys being calculated for each branch path.

It is an object of the present invention to provide a microprogrammed controller that is self-verifying in which valid branches along the execution path are verified and invalid branches are detected and isolated.

Applicants submit that this section of Detlefs teaches a “checker to report specific error messages.” (Page 29, Section 6, paragraph 2, lines 1-2). But Detlefs does not teach or suggest an error message that includes a program trace that identifies a path through the computer program.

Chan teaches a controller that is self-verifying. Specifically, valid branches along the execution path are verified. Invalid branches along the execution path are detected

and isolated. These sections of Chan, however, do not teach or suggest the claimed recitations. Claim 1 specifically recites: “program trace **that identifies a path through the computer program** when the counter-example identifies one or more of the flow control labels.”

Further, the combination of Detlefs and Chan fails to teach or suggest an error message that includes a program trace that identifies a **path through** the computer program. Detlefs merely teaches to report or label a specific error messages. Detlefs later discusses that an exact single location of the error message is reported (See page 29, Section 6). Chan merely teaches to detect and isolate invalid branches. Together, Detlefs and Chan, for example, would teach or suggest labeling an isolated invalid branch. By contrast, claim 1 recites an error message that includes a program trace that identifies a path through the computer program.

Detlefs and/or Chan not teach/suggest Flow Control Label Limitation

Claim 1 recites: “inserting flow control labels into the sub-equations of the logical equation, the flow control labels identifying conditional branch points in the specified computer program.” This limitation is not taught or suggested in Detlefs and/or Chan.

The Examiner **admits** that “Detlefs does not expressly disclose the limitation wherein the converting step includes inserting flow control labels into the sub-equation of the logical equation, the flow control labels identifying conditional branch points in the specified computer program.” (See Final OA, pages 4-5). The Examiner states that “Detlefs does show that any sub-equation of the logical equation can be given a label (see page 29, section 6, paragraph 2, lines 1-2).” (See Final OA, page 4). The Examiner further states, though, that Chan discloses this limitation (see column 1, lines 26-35 and 38-53).

Applicants agree with the Examiner that Detlefs does not expressly disclose the limitation wherein the converting step includes inserting flow control labels into the sub-equation of the logical equation, the flow control labels identifying conditional branch points in the specified computer program. Applicants contend, though, that the cited recitation in claim 1 is not disclosed or suggested anywhere in Detlefs and/or Chan.

The Examiner cites Detlefs (at page 29, section 6, paragraph 2, lines 1-2) and Chan (Col. 1, lines 26-35 and 38-53). This section of Detlefs teaches a static checker that reports the location of specific error message. In other words, “the label encodes the source position and error type.” (Id.). This section of Chan teaches a marker that “is computed **at runtime** and is matched with the stored marker to detect any wild branches.” (Emphasis added. Col. 1, lines 41-42). The combination thus teaches (1) reporting the location of a specific error message plus (2) computing a marker at runtime to detect wild branches. In other words, this combination teaches and suggests a dynamic checker that, at runtime, reports the location of specific error messages. This combination, however, does not teach or suggest the claimed limitation. Claim 1 specifically recites “inserting flow control labels into the sub-equations of the logical equation, the flow control labels identifying conditional branch points in the specified computer program.”

Detlefs and/or Chan not teach/suggest Theorem Prover Limitation

Claim 1 recites applying a theorem prover to the logical equation to determine the truth of the logical equation. When the truth of the logical equation cannot be proved, then “generating ... any of the flow control labels for conditional branch points of the program associated with the identified variable values.” This limitation is not taught or suggested in Detlefs and/or Chan.

The Examiner contends that Detlefs teaches a theorem prover with any of the flow control labels for conditional branch points of the program associated with the identified variable values. The Examiner cites Detlefs at Page 23, section 4, paragraph 1, lines 4-8. Applicants have scrutinized this section and find no such teaching. This section of Detlefs is reproduced below:

The condition is submitted to an automatic theorem-prover, just like in program verification, but unlike in program verification, we have no interest in the case where the theorem prover succeeds. Instead, the tool post-processes theorem-prover failures into meaningful error messages.

In short, this section plainly states that a condition is submitted to a theorem-prover. The program then processes failures into error messages. Detlefs later discusses that an exact location of these error messages is reported. (See page 29, section 6). This section, however, does not teach or suggest “flow control labels for conditional branch points of the program associated with the identified variable values.”

Conclusion Group I:

For at least the foregoing reasons, claim 1 is patentable over Detlefs in view of Chan. Independent claim 23 is patentable over Detlefs in view of Chan for at least the reasons given in connection with claim 1. As such, Applicants respectfully request reversal of the rejection of independent claims 1 and 23.

Dependent claims 2 - 22 and 24 - 46 depend from independent claims 1 and 23. Thus, for at least the reasons given in connection with claims 1 and 23, claims 2 - 22 and 24 - 46 are patentable over Detlefs in view of Chan. As such, Applicants respectfully request reversal of the rejection of dependent claims 2 - 22 and 24 - 46.

Applicants note that numerous dependent claims are rejected under § 103 as being unpatentable over Detlets in view of Chan, and further in view of Rickel. Applicants acknowledge that Rickel is generally directed to a debugging tool for debugging a binary program file. (See Abstract). Rickel, however, does not cure the deficiencies of Detlefs and Chan. As such, Applicants reiterate the arguments above with claim 1 with respect to all claims rejected under the combination of Detlefs, Chan, and Rickel.

B. GROUP II (Claim 47):

Claim 47 is representative of Group II. Claim 47 is rejected under 35 USC § 103(a) as being unpatentable over Detlefs in view of Chan.

To establish a prima facie case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art cited must teach or suggest all the claim

limitations. *See* M.P.E.P. § 2143. Applicants assert that the rejection does not satisfy these criteria.

Detlefs and/or Chan not teach/suggest Tracing Path Limitation

Claim 47 recites converting at least one counter-example into an error message. The error message “comprises a program trace that identifies a path through the computer program when the counter-example identifies one or more of the flow control labels.” This limitation is not taught or suggested in Detlefs and/or Chan.

The Examiner states that Detlefs teaches an error message that includes a **program trace that identifies a path through the computer program** when the counter-example identifies one or more of the flow control labels. The Examiner cites Detlef (Page 29, section 6, paragraph 2, lines 3-4 and 7-9) and Chan (Col. 1, lines 26-35).

This section of Detlef is reproduced below:

If the prover finds a counterexample, it emits the set of labels of relevant subformulas that are false in the counterexample. The name of the label encodes the source position and error type. This makes it straightforward to translate failed proofs into specific error messages.

This section of Chan is reproduced below:

One of the problems that arises in trying to check entire microcode sequences is checking the execution of conditional branching instructions. This is because branches outside the path taken in calculating the original check key will result in different check keys being calculated for each branch path.

It is an object of the present invention to provide a microprogrammed controller that is self-verifying in which valid branches along the execution path are verified and invalid branches are detected and isolated.

Applicants submit that this section of Detlefs teaches a “checker to report specific error messages.” (Page 29, Section 6, paragraph 2, lines 1-2). But Detlefs does not teach or suggest an error message that includes a program trace that identifies a path through the computer program.

Chan teaches a controller that is self-verifying. Specifically, valid branches along the execution path are verified. Invalid branches along the execution path are detected and isolated. These sections of Chan, however, do not teach or suggest the claimed recitations. Claim 47 specifically recites: “program trace **that identifies a path through the computer program** when the counter-example identifies one or more of the flow control labels.”

Further, the combination of Detlefs and Chan fails to teach or suggest an error message that includes a program trace that identifies a **path through** the computer program. Detlefs merely teaches to report or label a specific error messages. Detlefs later discusses that an exact single location of the error message is reported (See page 29, Section 6). Chan merely teaches to detect and isolate invalid branches. Together, Detlefs and Chan, for example, would teach or suggest labeling an isolated invalid branch. By contrast, claim 47 recites an error message that includes a program trace that identifies a path through the computer program.

Detlefs and/or Chan not teach/suggest Flow Control Labels Limitations

Claim 47 recites inserting flow control labels into sub-equations of the logical equation. The flow control labels identify branch points in the computer program. Claim 47 specifically recites that the flow control labels are “selected from the group consisting of (1) $L \Rightarrow P$ wherein L is a flow control label name and P is a subcomponent of the logical equation, (2) $L = K \Rightarrow P$ wherein L is a flow control label name, k is a constant value and P is a subcomponent of the logical equation, (3) $\{L \text{ LBLPOS } L \text{ } P\}$ wherein L is a flow control label name and P is a subcomponent of the logical equation, (4) $\neg\{L \text{ LBLNEG } L \neg P\}$ wherein L is a flow control label name and P is a subcomponent of the logical equation, and (5) $\{L \text{ LBLPOS } L \text{ True}\} \Rightarrow P$ wherein L is a flow control label name and P is a subcomponent of the logical equation.” These limitations are not taught or suggested in Detlefs and/or Chan.

Claim 47 recites specific examples of various flow control labels that are inserted into the sub-equations. These flow control labels are used to identify branch points in the computer program. The Examiner even **admits** that “Detlefs does not expressly disclose” the recited limitations of flow control labels. (See Final OA, page 12, lines 2 – 11).

To cure the admitted deficiency of Detlefs, the Examiner cites Chan at Col. 1, lines 26-35 and 38-53. Applicants have scrutinized these sections of Chan but cannot locate any teaching or suggestion of the recited limitations of flow control labels.

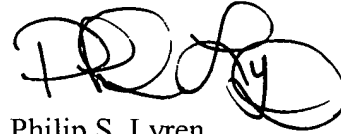
Conclusion Group II:

For at least the foregoing reasons, claim 47 is patentable over Detlefs in view of Chan. As such, Applicants respectfully request reversal of the rejection of independent claim 47.

CONCLUSION

In view of the above, Applicants respectfully request the Board of Appeals to reverse the Examiner's rejection of all pending claims 1 – 47.

Respectfully submitted,



Philip S. Lyren
Reg. No. 40,709
Ph: 281-514-8236

CERTIFICATE UNDER 37 C.F.R. 1.8: The undersigned hereby certifies that this paper or papers, as described herein, are being deposited in the United States Postal Service, as first class mail, in an envelope address to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on this 17th day of August 2004.

By

Name: Be Henry



APPENDIX TO THE APPEAL BRIEF UNDER 37 C.F.R. § 1.192

The Appendix is incorporated into the foregoing Appeal Brief under 37 C.F.R. 1.192.

THE CLAIMS

1. A method of verifying with static checking whether a specified computer program satisfies a predefined set of conditions, comprising:

converting the program into a logical equation representing the predefined set of conditions as applied to instructions and elements of the instructions of the program, the converting step including inserting flow control labels into the sub-equations of the logical equation, the flow control labels identifying conditional branch points in the specified computer program;

applying a theorem prover to the logical equation to determine the truth of the logical equation, and when the truth of the logical equation cannot be proved, generating at least one counter-example identifying one of the conditions, one or more variable values inconsistent with the one condition, and any of the flow control labels for conditional branch points of the program associated with the identified variable values; and

converting the at least one counter-example into an error message that includes a program trace that identifies a path through the computer program when the counter-example identifies one or more of the flow control labels.

2. The method of claim 1 wherein the converting step additionally comprises a step of converting the program into an intermediate language form prior to creating the

logical equation.

3. The method of claim 2 wherein the flow control labels are inserted before converting the program into the intermediate language form.

4. The method of claim 2 wherein the flow control labels are inserted after converting the program into the intermediate language form.

5. The method of claim 2 wherein the intermediate language form is Dijkstra's guarded command language.

6. The method of claim 1 wherein at least one of the flow control labels includes a flow control label name that includes a string that identifies a type of branch in the program and a line number in the specified computer program.

7. The method of claim 1 wherein at least one of the flow control labels includes a flow control label name that includes a value associated with an entry in a table that identifies a type of branch in the program and a line number in the specified computer program.

8. The method of claim 1 wherein at least one of the flow control labels is of the form $L \Rightarrow P$ wherein L is a flow control label name and P is a subcomponent of the logical equation.

9. The method of claim 8 wherein the flow control label name comprises a string that identifies a type of branch in the program and a line number in the specified computer program.

10. The method of claim 8 wherein the flow control label name includes a value associated with an entry in a table that identifies a type of branch in the program and a line number in the specified computer program.

11. The method of claim 1 wherein at least one of the flow control labels is of the form $L=K \implies P$ wherein L is a flow control label name, k is a constant value and P is a subcomponent of the logical equation.

12. The method of claim 11 wherein the flow control label name comprises a string that identifies a type of branch in the program and a line number in the specified computer program.

13. The method of claim 11 wherein the flow control label name includes a value associated with an entry in a table that identifies a type of branch in the program and a line number in the specified computer program.

14. The method of claim 1 wherein at least one of the flow control labels is of the form $\{L, LPOS, L, P\}$ wherein L is a flow control label name and P is a subcomponent

of the logical equation.

15. The method of claim 14 wherein the flow control label name comprises a string that identifies a type of branch in the program and a line number in the specified computer program.

16. The method of claim 14 wherein the flow control label name includes a value associated with an entry in a table that identifies a type of branch in the program and a line number in the specified computer program.

17. The method of claim 1 wherein at least one of the flow control labels is of the form $\neg\{\text{LBLNEG } L \neg P\}$ wherein L is a flow control label name and P is a subcomponent of the logical equation.

18. The method of claim 17 wherein the flow control label name comprises a string that identifies a type of branch in the program and a line number in the specified computer program.

19. The method of claim 17 wherein the flow control label name includes a value associated with an entry in a table that identifies a type of branch in the program and a line number in the specified computer program.

20. The method of claim 1 wherein at least one of the flow control labels is of

the form $\{LBLPOS\ L\ True\} \implies P$ wherein L is a flow control label name and P is a subcomponent of the logical equation.

21. The method of claim 20 wherein the flow control label name comprises a string that identifies a type of branch in the program and a line number in the specified computer program.

22. The method of claim 1 wherein the flow control label name identifies a line number in the specified computer program at which an associated program instruction is located and includes a sequence number indicating an order of execution of the program instruction at the identified line number relative to other program instructions identified by other flow control labels.

23. A computer program product for use in static checking in conjunction with a computer system, the computer program product comprising a computer readable storage medium and a computer program mechanism embedded therein, the computer program mechanism comprising:

a verification condition generator that converts a specified program into a logical equation representing the predefined set of conditions as applied to instructions and elements of the instructions of the program, the verification condition generator including instructions that insert flow control labels into the sub-equations of the logical equation, the flow control labels identifying conditional branch points in the specified computer program;

a theorem prover that processes the logical equation to determine the truth of the logical equation, and when the truth of the logical equation cannot be proved, generates at least one counter-example identifying one of the conditions, one or more variable values inconsistent with the one condition, and any of the flow control labels for conditional branch points of the program associated with the identified variable values; and

a post processing module that converts the at least one counter-example into an error message that includes a program trace that identifies a path through the computer program when the counter-example identifies one or more of the flow control labels.

24. The computer program product of claim 23 wherein the verification condition generator includes instructions for converting the program into an intermediate language form prior to creating the logical equation.

25. The computer program product of claim 24 wherein the verification condition generator includes instructions for inserting the flow control labels before converting the program into the intermediate language form.

26. The computer program product of claim 24 wherein the verification condition generator includes instructions for inserting the flow control labels after converting the program into the intermediate language form.

27. The computer program product of claim 24 wherein the intermediate language form is Dijkstra's guarded command language.

28. The computer program product of claim 23 wherein at least one of the flow control labels includes a flow control label name that includes a string that identifies a type of branch in the program and a line number in the specified computer program.

29. The computer program product of claim 23 wherein at least one of the flow control labels includes a flow control label name that includes a value associated with an entry in a table that identifies a type of branch in the program and a line number in the specified computer program.

30. The computer program product of claim 23 wherein at least one of the flow control labels is of the form $L \Rightarrow P$ wherein L is a flow control label name and P is a subcomponent of the logical equation.

31. The computer program product of claim 30 wherein the flow control label name comprises a string that identifies a type of branch in the program and a line number in the specified computer program.

32. The computer program product of claim 30 wherein the flow control label name includes a value associated with an entry in a table that identifies a type of branch in the program and a line number in the specified computer program.

33. The computer program product of claim 23 wherein at least one of the

flow control labels is of the form $L=k \Rightarrow P$ wherein L is a flow control label name, k is a constant value and P is a subcomponent of the logical equation.

34. The computer program product of claim 23 wherein names of the flow control labels comprise a string that identifies a type of branch in the program and a line number in the specified computer program.

35. The computer program product of claim 23 wherein names of the flow control labels include a value associated with an entry in a table that identifies a type of branch in the program and a line number in the specified computer program.

36. The computer program product of claim 23 wherein at least one of the flow control labels is of the form $\{L \text{ LBLPOS } L \text{ } P\}$ wherein L is a flow control label name and P is a subcomponent of the logical equation.

37. The computer program product of claim 36 wherein the flow control label name comprises a string that identifies a type of branch in the program and a line number in the specified computer program.

38. The computer program product of claim 36 wherein the flow control label name includes a value associated with an entry in a table that identifies a type of branch in the program and a line number in the specified computer program.

39. The computer program product of claim 23 wherein at least one of the flow control labels is of the form $\neg\{\text{LBLNEG } L \neg P\}$ wherein L is a flow control label name and P is a subcomponent of the logical equation.

40. The computer program product of claim 39 wherein the flow control label name comprises a string that identifies a type of branch in the program and a line number in the specified computer program.

41. The computer program product of claim 39 wherein the flow control label name includes a value associated with an entry in a table that identifies a type of branch in the program and a line number in the specified computer program.

42. The computer program product of claim 23 wherein at least one of the flow control labels is of the form $\{\text{LBLPOS } L \text{ True}\} \Rightarrow P$ wherein L is a flow control label name and P is a subcomponent of the logical equation.

43. The computer program product of claim 42 wherein the flow control label name comprises a string that identifies a type of branch in the program and a line number in the specified computer program.

44. The computer program product of claim 42 wherein the flow control label name includes a value associated with an entry in a table that identifies a type of branch in the specified computer program and a line number in the specified computer program.

45. The computer program product of claim 23 wherein the flow control label name identifies a line number in the specified computer program at which an associated program instruction is located and includes a sequence number indicating an order of execution of the program instruction at the identified line number relative to other program instructions identified by other flow control labels.

46. The method of claim 20 wherein the flow control label name includes a value associated with an entry in a table that identifies a type of branch in the specified computer program and a line number in the specified computer program.

47. A method, comprising:
converting a computer program into a logical equation representing a predefined set of conditions;

inserting flow control labels into sub-equations of the logical equation, the flow control labels identifying branch points in the computer program, wherein at least one of the flow control labels is of a form selected from the group consisting of (1) $L \Rightarrow P$ wherein L is a flow control label name and P is a subcomponent of the logical equation, (2) $L = K \Rightarrow P$ wherein L is a flow control label name, k is a constant value and P is a subcomponent of the logical equation, (3) $\{LBLPOS L P\}$ wherein L is a flow control label name and P is a subcomponent of the logical equation, (4) $\neg\{LBLNEG L \neg P\}$ wherein L is a flow control label name and P is a subcomponent of the logical equation, and (5) $\{LBLPOS L True\} \Rightarrow P$ wherein L is a flow control label name and P is a

subcomponent of the logical equation;

applying a theorem prover to the logical equation to determine a truth of the logical equation, and when the truth of the logical equation cannot be proved, generating at least one counter-example identifying one of the conditions; and

converting the at least one counter-example into an error message that comprises a program trace that identifies a path through the computer program when the counter-example identifies one or more of the flow control labels.